

Keep it Simple: Pruning Papyrus(-RT) for Real-Time Systems

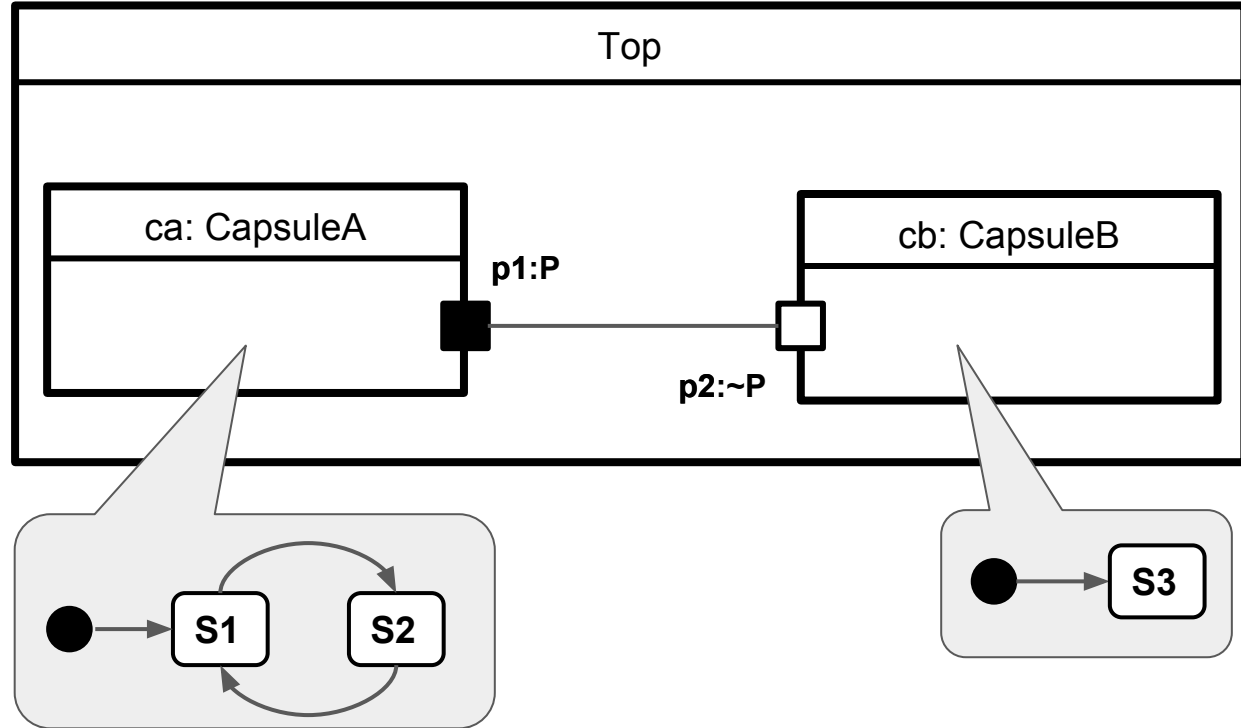
Nicolas Hili, Juergen Dingel

Sum Ergo
Computo
I am therefore I compute

Context

UML for Real-Time:

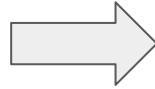
- Subset of UML2
- A small set of concepts:
 - Capsules
 - Capsule Parts
 - Ports
 - Protocols
 - Connectors
 - StateMachines
- Action semantics



Motivation

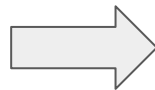


UML-RT is specific



Need for dedicated environments

Development of Real-Time
(Embedded) Systems



Support for specific real-time activities
Support for (configurable) platform modeling



Part I

Customizing Papyrus-(RT)

How and Where to Customize Papyrus ?

The screenshot displays the Eclipse SDK interface for the Papyrus UML editor. The title bar indicates the project is 'runtime-Rover - Java - PingPongCopy/pingPong.di - Eclipse SDK'. The main window is divided into several panes:

- Model Explorer (Left):** Shows the project structure. The 'RootElement' is expanded, showing a hierarchy of elements including 'PingPong', 'PingPongIO', 'Pinger', 'Ponger', and 'Ponger::PongerStateMachine'. Arrows point from this pane to the diagram and the Properties view.
- Diagram Area (Center):** Displays the UML diagram. The top diagram shows a 'Top' package with two classes: 'Pinger' and 'Ponger'. The 'Pinger' class has a field 'pinger: PingPong [1]' and a connector. The 'Ponger' class has a field 'ponger: ~PingPong [1]' and a connector. The bottom diagram shows the 'Ponger::PongerStateMachine' with an initial state 'init' leading to a state 'Playing' with the label '/entry OpaqueBehavior null'.
- Properties View (Bottom):** Shows the properties of the selected element, 'Ponger : Ponger'. The 'UML' tab is active, displaying various properties and their values.

The Properties View for 'Ponger : Ponger' includes the following sections:

- UML:** Name: ponger
- Comments:** Is derived: ☐ true ☒ false
- Profile:** Is leaf: ☐ true ☒ false
- Style:** Is read only: ☐ true ☒ false
- Appearance:** Is unique: ☒ true ☐ false
- Rulers And Grid:** Type: Ponger
- Advanced:** Is derived union: ☐ true ☒ false; Is ordered: ☒ true ☐ false; Is static: ☐ true ☒ false; Visibility: protected; Multiplicity: 1

Papyrus-RT

runtime-Rover - Java - PingPong/pingPong.di - Eclipse SDK

File Edit Diagram Navigate Search Papyrus Project UML-RT Menu Run Window Help

Model Explorer

- RootElement
 - «Protocol» PingPong
 - out ping ()
 - in pong ()
 - «Capsule» Pinger
 - «RTPort» pinger : PingPong
 - «RTStateMachine» StateMachine
 - «Capsule» Ponger
 - «RTPort» ponger : ~PingPong
 - «RTStateMachine» PongerStateMachine
 - «RTRegion» Region
 - init
 - «RTPseudostate» <Pseudostate>
 - «RTState» Playing
 - <Opaque Behavior>
 - Ponger
 - «Capsule» Top
 - «CapsulePart» pinger : Pinger
 - «CapsulePart» ponger : Ponger
 - «RTConnector» connector

«EPackage, ModelLibrary» UML Primitive Types

«ModelLibrary» Ecore Primitive Types

pingPong.di

Top

pinger: Pinger [1]
+ pinger: PingPong [1]

«RTConnector»
connector

ponger: Ponger [1]
+ ponger: ~PingPong [1]

PongerStateMachine

init

Playing
/entry OpaqueBehavior
null

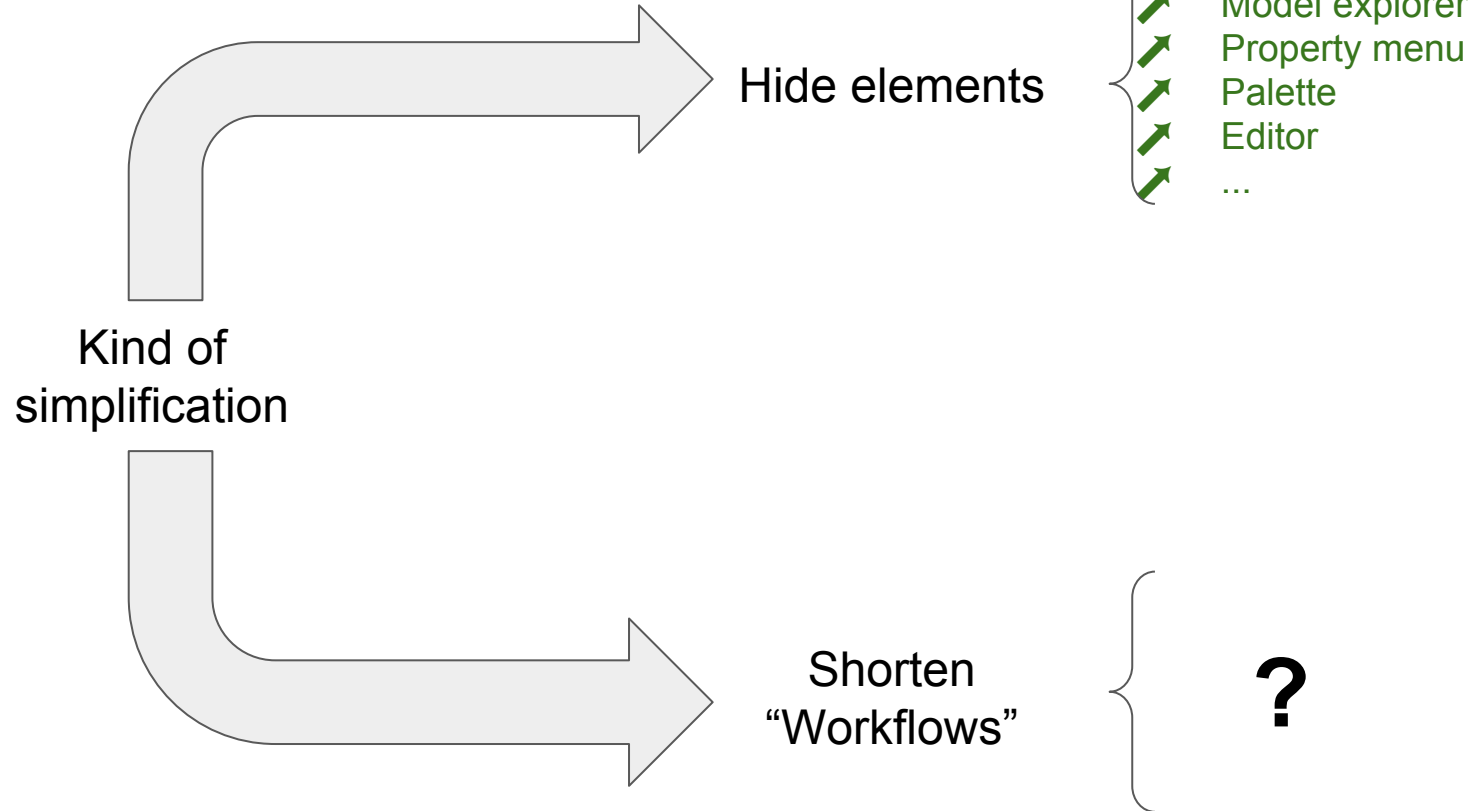
Ponger::PongerStateMachine

Problems Javadoc Declaration Properties

ponger : Ponger

UML-RT	Name	ponger
UML	Capsule	«Capsule» Ponger
Comments		
Profile	Capsule Part Properties	
Style	Kind	<input checked="" type="radio"/> Fixed <input type="radio"/> Optional <input type="radio"/> Plugin
Appearance	Multiplicity	1
Rulers And Grid	Aggregation	composite

Simplifying Papyrus



Shortening “Workflows”: Element creation

runtime-Rover - Java - Papyrus

File Edit Diagram Navigate Search Papyrus Project UML-RT Menu Run

Model Explorer

- RootElement
 - «Protocol» PingPong
 - «Capsule» Pinger
 - «RTPort» pinger : PingPong
 - «RTStateMachine» PingerStateMachin
 - Pinger
 - «Capsule» Ponger
 - «RTPort» pingPong : ~PingPong
 - «RTStateMachine» PongerStateMachir
 - Ponger
 - «Capsule» Top
 - «CapsulePart» pinger : Pinger
 - Top

pingPong.di

Top

pinger: Pinger [1]
+ pinger: PingPong [1]

Palette

- Port
- Capsule Part
- Connector

Problems @ Javadoc Declaration Properties

«Capsule» Ponger

UML-RT	Name
UML	Ponger
Comments	
Profile	
Advanced	

1 Item selected

DnD the **Ponger** capsule into the **Top** capsule...

... Will automatically initialize the **ponger** capsule part.

Shortening “Workflows”: Diagram Initialization & Navigation

The screenshot displays the Eclipse IDE interface with the title bar "runtime-Rover - Java - PingPong/pingPong.di - Eclipse SDK". The menu bar includes File, Edit, Diagram, Navigate, Search, Papyrus, Project, UML-RT Menu, Run, Window, and Help. The left sidebar contains the Model Explorer, showing a tree structure with elements like RootElement, «Protocol» PingPong, «Capsule» Pinger, «RTPort» pinger : PingPong, «RTStateMachine» PingerStateMachin, Pinger, «Capsule» Ponger, «RTPort» pingPong : ~PingPong, «RTStateMachine» PongerStateMachir, Ponger, «Capsule» Top, «CapsulePart» pinger : Pinger, «CapsulePart» ponger : Ponger (highlighted), and Top. The main diagram area shows a UML diagram with two capsule parts. A callout box points to the newly created capsule part with the text "Double-clicking on the newly created capsule part...". Another callout box points to the capsule part with the text "Double-clicking on the capsule does nothing, though.". A third callout box points to the diagram with the text "... Will automatically open the Ponger diagram (automatically initialized when the Ponger capsule was created).". The right sidebar contains the Palette, showing elements like Port, Capsule Part, and Connector. The bottom of the IDE shows the Problems, Declaration, Properties, and UML-RT tabs.

Double-clicking on the newly created capsule part...

Double-clicking on the capsule does nothing, though.

... Will automatically open the Ponger diagram (automatically initialized when the Ponger capsule was created).

Shortening “Workflows”: StateMachine modeling

The screenshot displays the Eclipse IDE interface for modeling a StateMachine. The title bar indicates the project is 'runtime-Rover - Java - PingPong/pingPong.di - Eclipse SDK'. The left-hand 'Model Explorer' shows a hierarchical tree of the model, including 'RootElement', '«Protocol» PingPong', and various '«Capsule»' components like 'Pinger' and 'Ponger'. A callout points to the 'Pinger' capsule, stating: 'The language has to be locally specified...'. The main editor shows a diagram of a capsule with a callout pointing to its internal code: '# pinger: Pinger [1]' and '+ pinger: PingPong [1]', with the text: 'Before defining the implementation'. Below the diagram, the 'Properties' view shows the 'Language' dropdown set to 'C++', with a callout stating: 'Defining the language can be done globally...'. At the bottom, the 'UML-RT' view shows the 'Language' dropdown set to 'UML', with a callout pointing to it: 'But every time a Behavior has to be created...'. The right-hand 'Palette' shows available components like 'Port', 'Capsule Part', and 'Connector'.

runtime-Rover - Java - PingPong/pingPong.di - Eclipse SDK

File Edit Diagram Window Help

Model Explorer

RootElement

- «Protocol» PingPong
- «Capsule» Pinger
 - «RTPort» pinger : PingPong
 - «RTStateMachine» PingerStateMachine
 - Pinger
- «Capsule» Ponger
 - «RTPort» pingPong : ~PingPong
 - «RTStateMachine» PongerStateMachine
 - «RTRegion» Region
 - Init
 - «RTPseudostate» <Pseudostate>
 - «RTState» Playing
 - Ponger::PongerStateMachine
 - Ponger
 - «Capsule» Top
 - «CapsulePart» pinger : Pinger
 - «CapsulePart» ponger : Ponger
 - Top
- «EPackage, ModelLibrary» UML Primitive Types
- «ModelLibrary» Ecore Primitive Types

Top

pinger: Pinger [1]
+ pinger: PingPong [1]

Before defining the implementation

Defining the language can be done globally...

RootElement

UML-RT Language C++

Language

UML

Comments

Profile

But every time a Behavior has to be created...

Palette

- Capsule St...
- Port
- Capsule Part
- Connector

Shortening “Workflows”: Behavior modeling

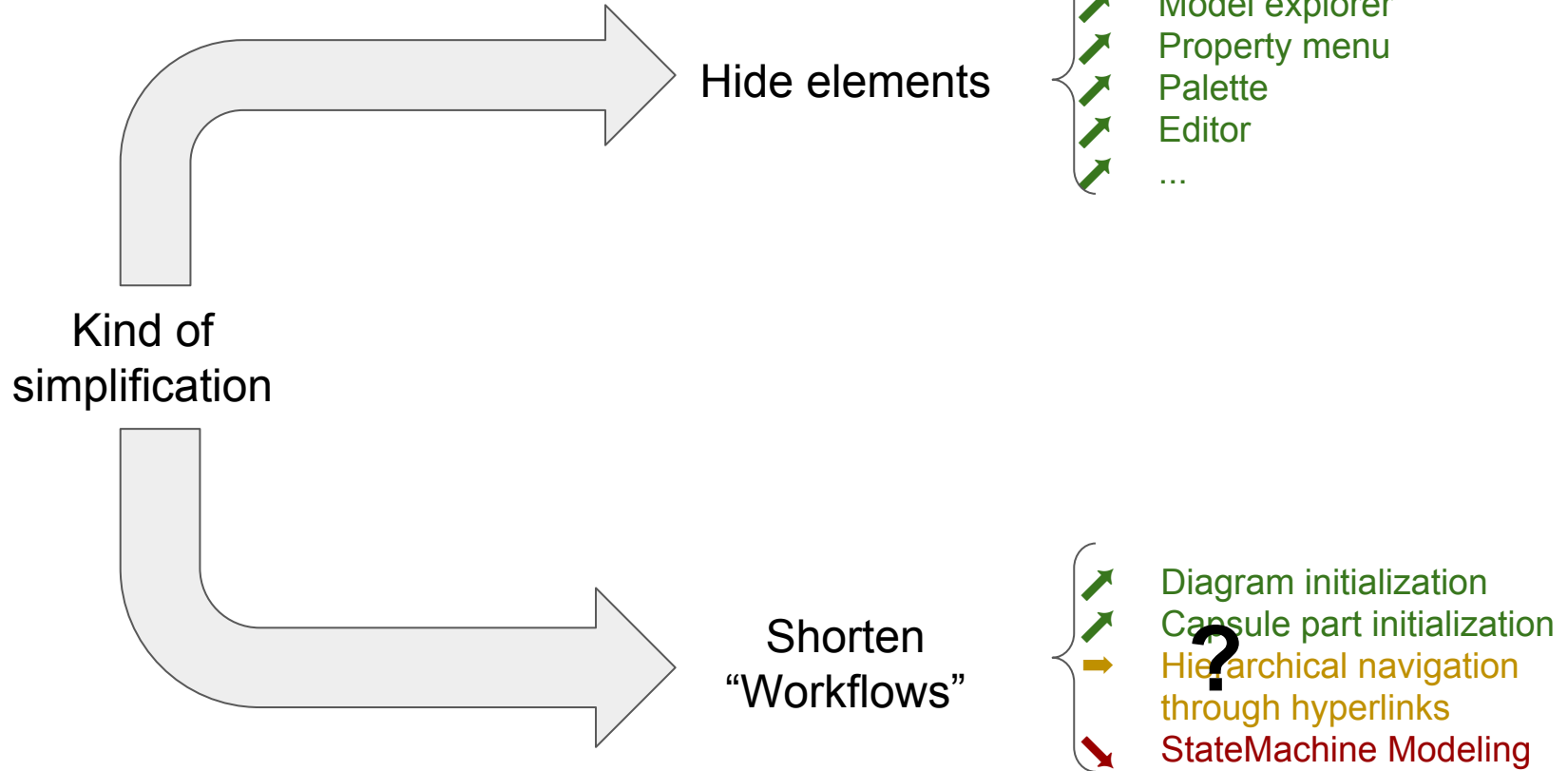
The screenshot displays the Eclipse IDE interface for a project named 'runtime-Rover - Java - PingPong/pingPong.di'. The main editor shows a UML diagram titled 'PongerStateMachine' with an initial state leading to a state named 'Playing'. The left-hand 'Model Explorer' shows a hierarchical structure of the model, including 'RootElement', 'Protocol PingPong', 'Capsule Pinger', and various 'RTState' and 'RTRegion' elements. The 'init' state is selected in the explorer.

Two dialog boxes are open in the foreground:

- Create a new Constraint**: This dialog has fields for 'Name', 'Visibility' (set to 'public'), 'Context' (set to '<UML>'), and 'Specification' (set to '<UML>'). It also includes a 'Constrained element' field with selection icons.
- Create a new OpaqueExpression**: This dialog has fields for 'Name', 'Language' (set to 'C++'), 'Visibility' (set to 'public'), 'Behavior' (set to '<Undefined>'), and 'Type' (set to '<Undefined>'). It also includes a 'Constrained element' field with selection icons.

A callout box points to the 'Constrained element' field in the 'Create a new Constraint' dialog, containing the text: "The same goes for defining constraints".

Simplifying Papyrus



Shortening “Workflows”: Making some Assumptions

Assumption 1:

A **capsule** may or may not have a **behavior**. If a behavior is defined, then it is modeled using a **StateMachine**.

Assumption 2:

While UML supports different behaviors (e.g., **Activity**, **Function Behavior**, **Opaque Behavior**), for defining **entry** / **exit** / **do** activities and **transition's effect**, only **Opaque Behaviors** are used by Papyrus-RT.

Assumption 3:

If the **language** is globally defined, there is no need to locally redefine it when creating the **opaque behaviors**.

Assumption 4:

If an **opaque behavior** can be defined in the target language (e.g., C++), the action semantics is defined by the Papyrus **RunTime Service**.

Shortening “Workflows”: Opaque Behavior modeling

runtime-Rover - Java - PingPong/pingPong.di - Eclipse SDK

File Edit Diagram Navigate Search Papyrus Project UML-RT Menu Run Window Help

Model Explorer

Create a new OpaqueBehavior

Name

Language

C++

Is abstract

Is leaf

Visibility

Specification

Use case

UML Comments

Cancel OK

port

Outgoing message

std::cout << "hello world" << std::endl;

While anything can be written in this box...

...most of the time, the action semantics is limited to sending messages to other capsules.

Palette

States and...

State

Transition

Choice

Deep History

Entry Point

Exit Point

Initial State

Properties

Playing

<Undefined>

Entry

<Undefined>

Exit

<Undefined>

Shortening “Workflows”: Contributions

Assumption 1:

A **capsule** may or may not have a **behavior**. If a behavior is defined, then it is modeled using a **StateMachine**.

Assumption 2:

While UML supports different behaviors (e.g., **Activity**, **Function Behavior**, **Opaque Behavior**), for defining **entry / exit / do** activities and **transition's effect**, only **Opaque Behaviors** are used by Papyrus-RT.

Assumption 3:

If the **language** is globally defined, there is no need to locally redefine it when creating the **opaque behaviors**.

Assumption 4:

If an **opaque behavior** can be defined in the target language (e.g., C++), the action semantics is defined by the Papyrus **RunTime Service**.

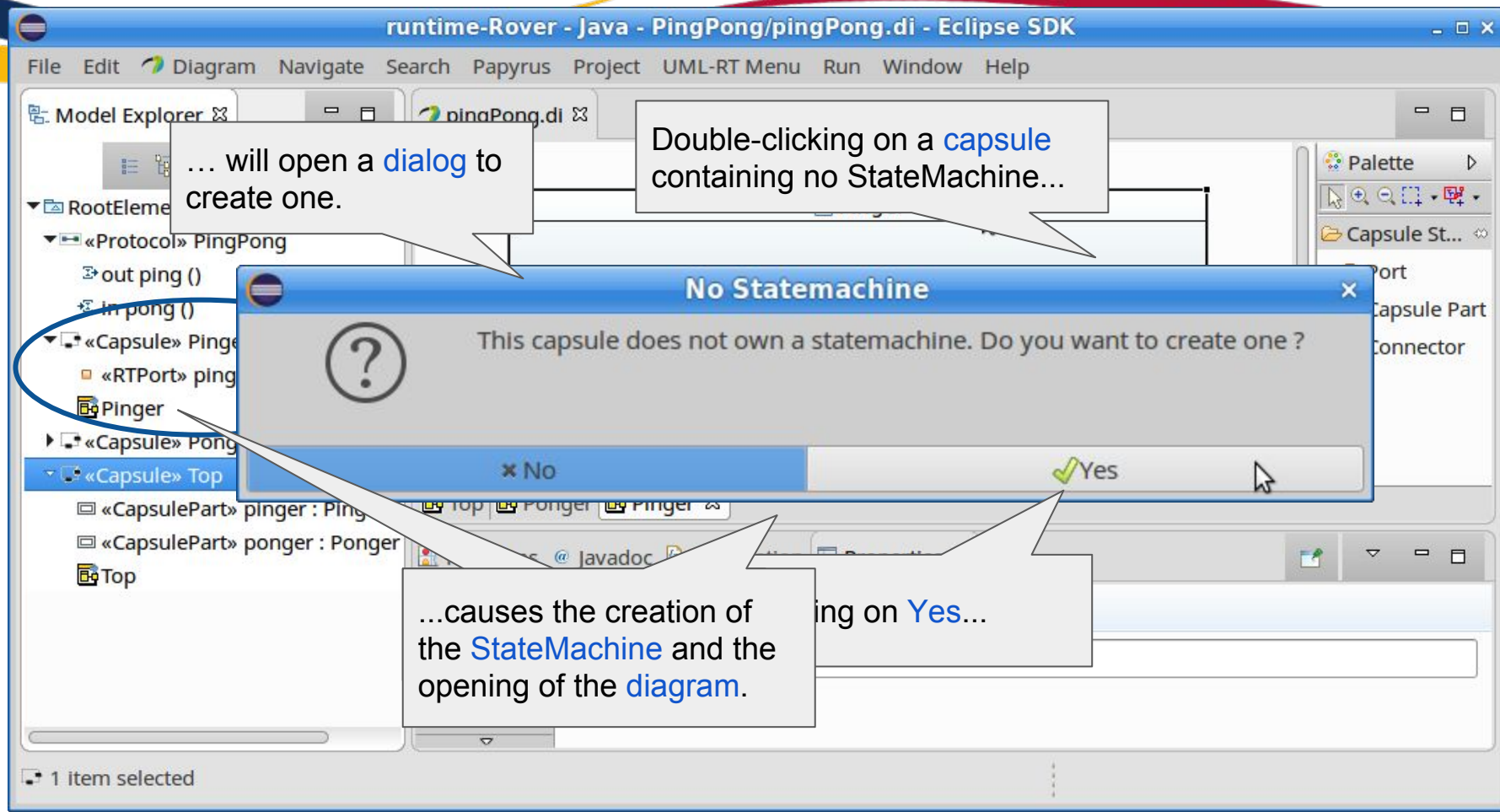
Customization 1:

- Initializing StateMachine models
- Extending the Hyperlink feature to open StateMachine diagrams

Customization 2:

- Simple hybrid textual / graphical notation for modeling state and transition behaviors.

Shortening “Workflows”: StateMachine initialization



Shortening “Workflows”: StateMachine initialization

The screenshot displays the Eclipse IDE interface with the following components:

- Model Explorer (Left):** Shows a project structure starting with `RootElement`, followed by `«Protocol» PingPong` (containing `out ping ()` and `In pong ()`). Under `PingPong` is a `«Capsule» Pinger`, which contains a `«Capsule» Ponger`. The `Ponger` capsule contains an `«RTPort» pingPong : ~PingPong [1]` and an `«RTStateMachine» PongerState`. Below this is a `«Capsule» Top` containing `«CapsulePart» pinger : Pinger`, `«CapsulePart» ponger : Ponger`, and `Top`.
- Diagram Editor (Center):** Displays the `*pingPong.di` diagram. It features a large capsule labeled `Ponger`. Inside the capsule, on the left, is a port labeled `+ pingPong: ~PingPong [1]`. A mouse cursor is positioned over the top-right corner of the `Ponger` capsule. A callout bubble points to this corner with the text: "Double-clicking on a capsule that already contains a StateMachine...".
- Palette (Right):** Shows the `Capsule St...` palette with elements: `Port`, `Capsule Part`, and `Connector`.
- Diagram Tabs (Bottom):** Includes tabs for `Top`, `Ponger` (selected), `Pinger`, and `Pinger::StateMachine`.
- Properties View (Bottom):** Shows the `«Capsule» Ponger` selected. It has tabs for `UML-RT` and `UML`. The `UML-RT` tab is active, showing a table with the header `Name` and one empty row.
- Callout Bubble (Bottom):** A speech bubble points to the `UML-RT` tab with the text: "...causes the opening of the existing diagram."
- Status Bar (Bottom Left):** Indicates "1 item selected".

Shortening “Workflows”: Hybrid Notation

runtime-Rover - Java - PingPong/pingPong.di - Eclipse SDK

File Edit Diagram Navigate Search Papyrus Project UML RT Menu Run Window Help

Model Explorer

- RootElement
 - «Protocol» PingPong
 - «Capsule» Pinger
 - «Capsule» Ponger
 - «RTPort» ponger :
 - «RTStateMachine»
 - «RTRegion» Region
 - Init
 - «RTPseudostate» <Pseudost...
 - «RTState» Playing
 - Ponger::PongerStateMachine
 - Ponger
 - «Capsule» Top

...and displayed.

When selecting a **state** or a **transition**, the Action view's **embedded editor** becomes active.

...the **UML artifacts** are **created**...

Autocompletion of **ports**, **incoming** and **outgoing messages**.

It contains an **Xtext embedded editor** for editing **state** and **transition behaviors**.

When the editor contents is **saved**...

When a **state** is selected, the editor supports the creation of **entry / exit / do actions**, as well as **internal transitions**.

Transition

- Choice
- Deep History
- Entry Point
- Exit Point
- Initial State
- Junction

Declaration Properties

class is : class org.eclipse.papy

Shortening “Workflows”: Hybrid Notation

runtime-Rover - Java - PingPong/pingPong.di - Eclipse SDK

File Edit Diagram Navigate Search Papyrus Project UML-RT Menu Run Window Help

Model Explorer

- RootElement
 - «Protocol» PingPong
 - «Capsule» Pinger
 - «Capsule» Ponger
 - «RTPort» ponger : ~PingPong
 - «RTStateMachine» PongerStateM
 - «RTRegion» Region
 - Init
 - <Transition>
 - <Opaque Behavior>
 - <Trigger>
 - «RTPseudostate» <Pseudost
 - «RTState» Playing
 - <Opaque Behavior>
 - <Region>

pingPong.di

PongerStateMachine

init

Playing

/entry OpaqueBehavior null

Palette

- States and...
- State
- Transition
- Choice
- Deep History
- Entry Point
- Exit Point
- Initial State
- Junction

Top Ponger Pinger Pinger::StateMachine Ponger::PongerStateMachine

Problems Javadoc Declaration Properties Action View

Current single selection class is : class org.eclipse.papyrus.uml.diagram.statemachine.custom.edit.part.CustomTr

```
on ponger.ping()/
ponger.pong().send();
```

The same goes for external transitions.

Shortening “Workflows”: Hybrid Notation



Hybrid textual / graphical notation:


- Implementation of a specific view ;
- Integration of an embedded Xtext editor ;
- Supports state's entry / do / exit actions ;
- Supports state's internal transition ;
- Supports external transition's effect.

Current limitations:

- Basic action semantics (e.g., does not support constraints, or multiple triggers) ;
- Does not support complex behaviors (e.g., written in C++).

Work to do:

- Improve the grammar ;
- Support of the Papyrus RTS library (log, frame, timer) ;
- Mixing with other Xtext grammars (e.g., C++) when the target language is defined.

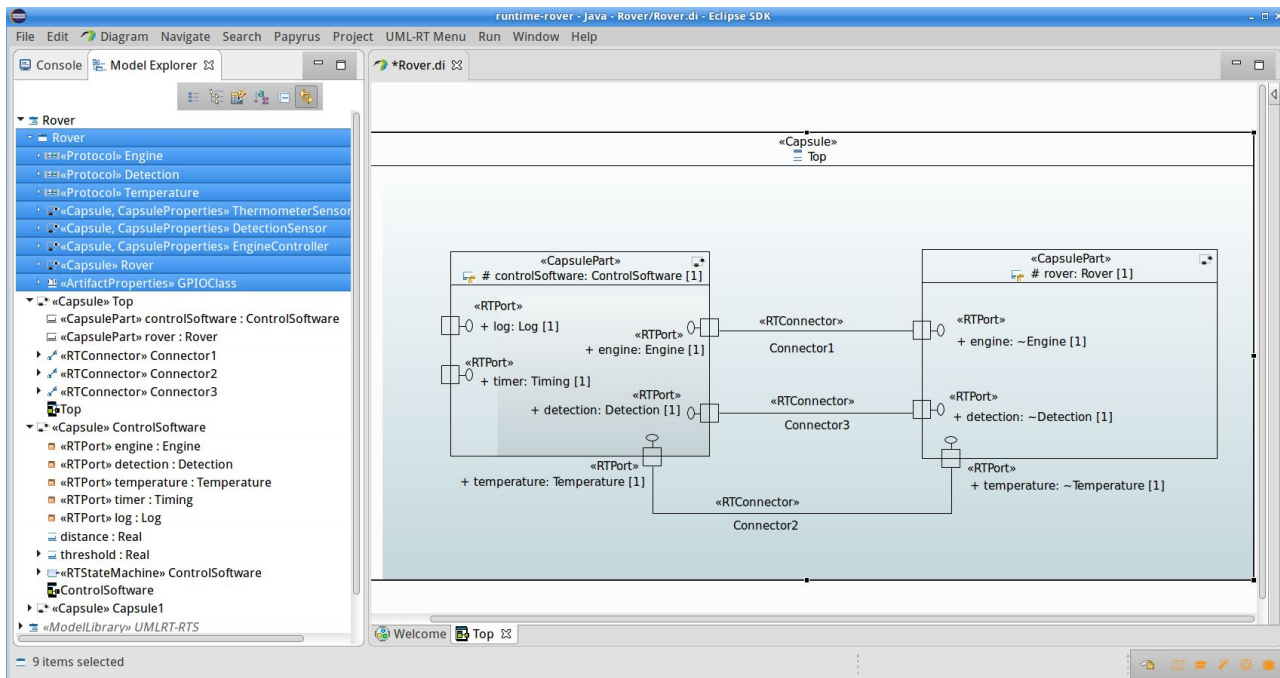


Part II

Supporting Specific Activities for Real-time (Embedded) System Development

The Rover Case Study

Graphical Model using Papyrus-RT



Textual Model in TUML-RT

```
model Model
{
    import uri "platform:/resource/org.eclipse.papyrusrtd.umlrt.common"
    import uri "platform:/resource/org.eclipse.papyrusrtd.umlrt.common"

    artifact GPIO {
    }

    capsule ControlSoftware {
        conjugate port engine: Rover.Engine;
        conjugate port detection: Rover.Detection;
        conjugate port temperature: Rover.Temperature;
        port timer: RTSLibrary.Timing;
        port log: RTSLibrary.Log;

        statemachine
        {
            initial controlSoftware_init;

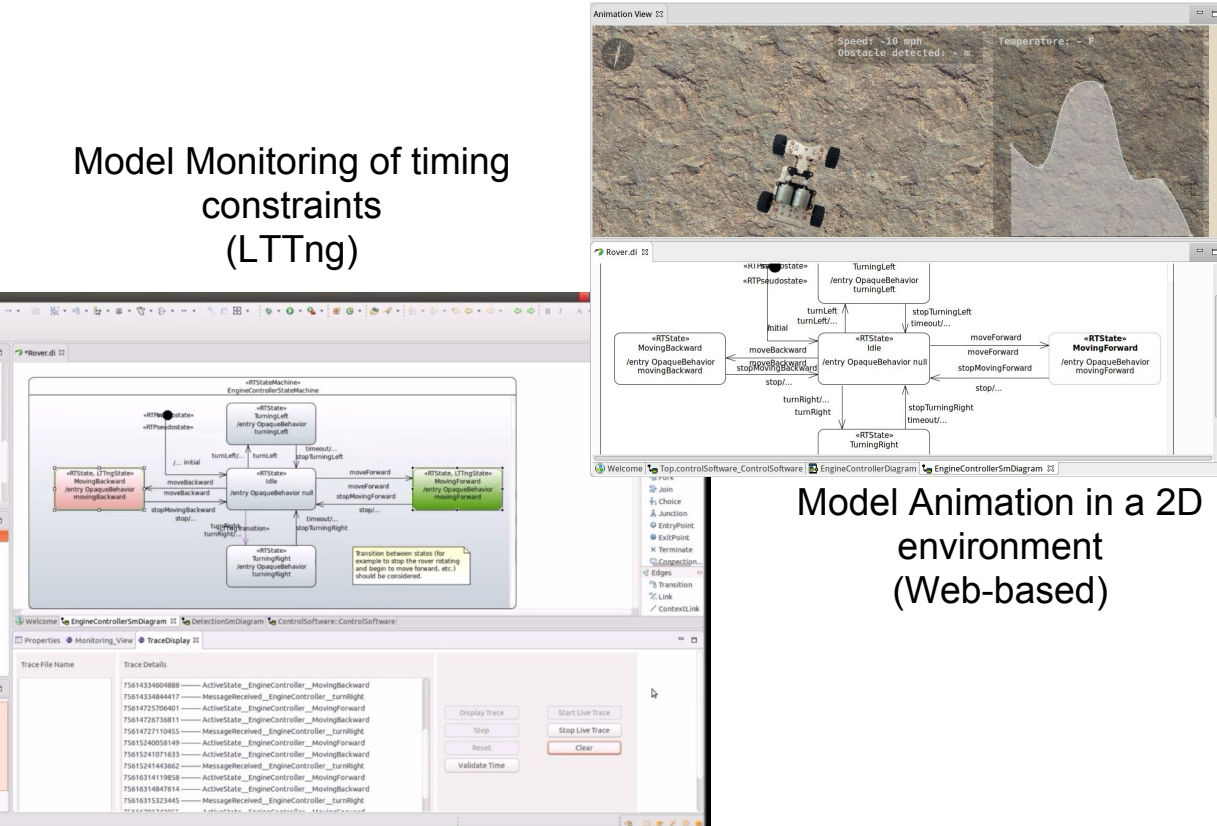
            state IDLE
            {
                entry action '
                    timer.informIn(UMLRTTimespec( 3, 0 ));
                    log.log("Entering into IDLE State");
                '
            }

            state MOVING_FOWARDS
            {
                entry action '
                    engine.moveForward().send();
                    timer.informIn(UMLRTTimespec( 3, 0 ));
                    log.log("Entering into Moving Forward State");
                '
            }

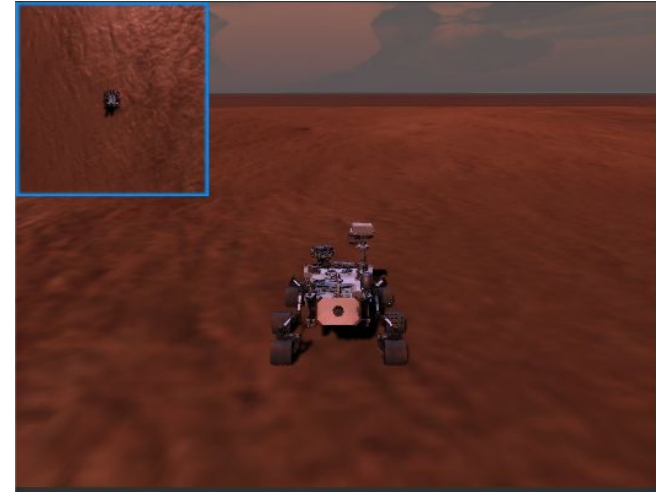
            state MOVING_BACKWARDS
            {
                entry action '
                    engine.stop().send();
                    engine.moveBackwards().send();
                    timer.informIn(UMLRTTimespec( 3, 0 ));
                    log.log("Entering into Moving Backwards State");
                '
            }
        }
    }
}
```


Supporting Specific Activities for RTE Systems

Model Monitoring of timing constraints (LTTng)

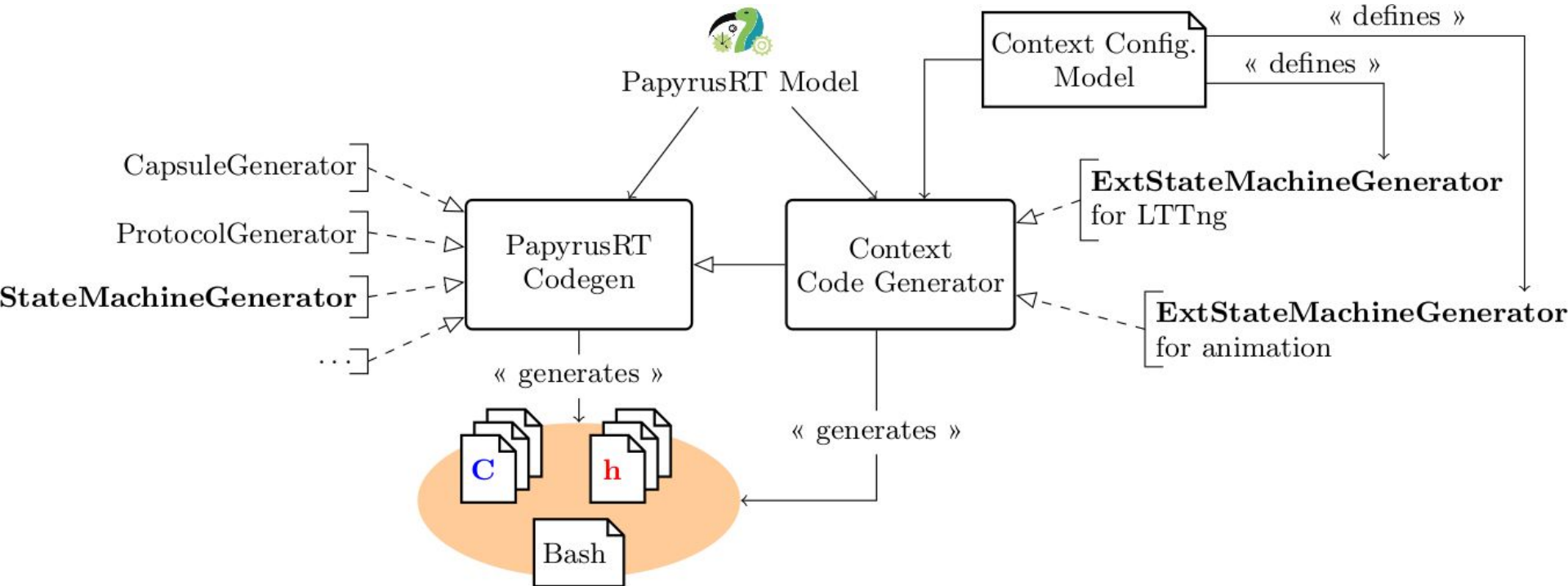


Model Animation in a 2D environment (Web-based)



Model Animation in a 3D environment (Unity)

Extending the PapyrusRT code generator



Platform Layering

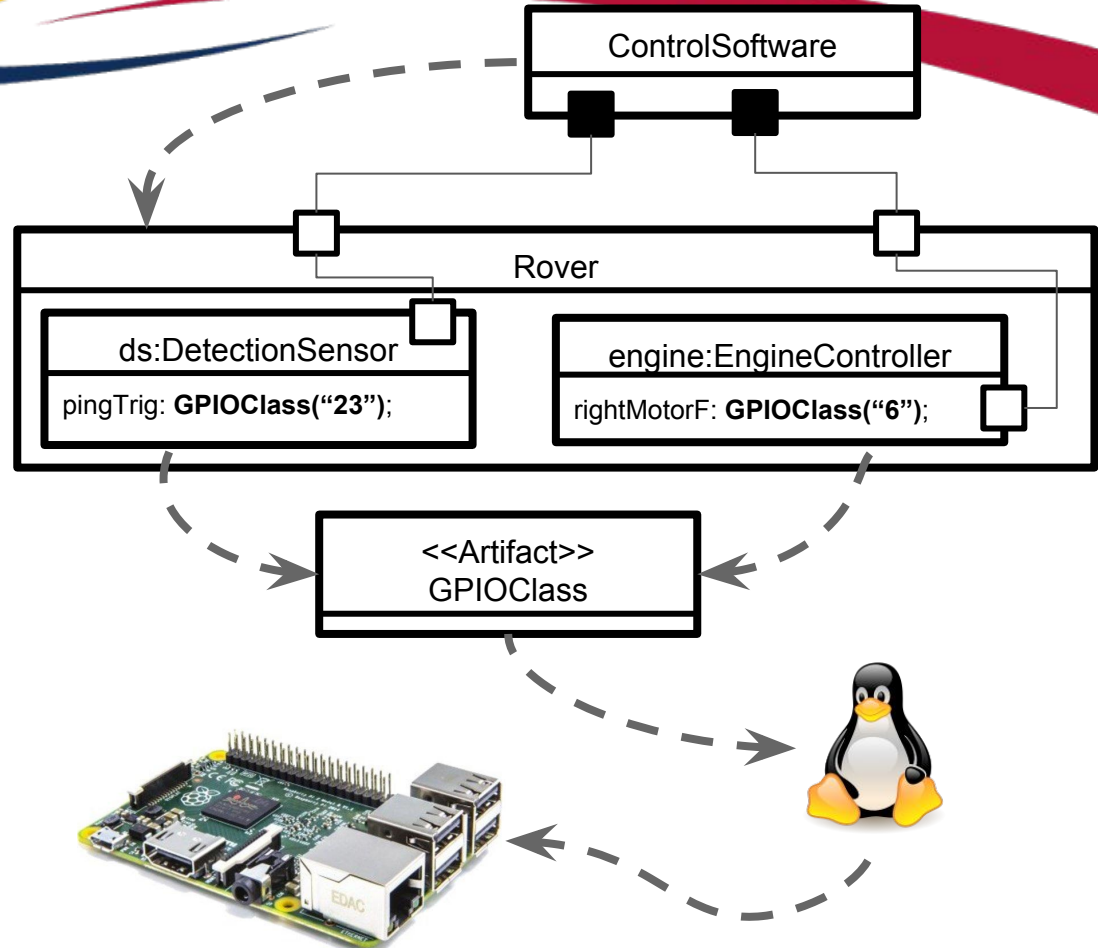
Application

Rover Library

GPIO Class

File System

Hardware



Platform Layering

Application

- Contains the Business Logic ;
- Does not know about the HW configuration ;
- Interacts with the Rover Library.

Rover Library

- Define the protocols the Business Logic will have to interact with ;
- Interacts with the Hardware ;
- **Specific to a design configuration.**

GPIO Class

File System

Hardware

? Problem:
How to change the design configuration without having to modify the Rover Library ?

✓ Proposition:
Embedding a specific configuration page with the loaded library

Allow for editing the design configuration (i.e., the GPIO mapping)

The UMLRT-Rover package is loaded in the model.

Give an overview of the platform.

Allow for editing the design configuration (i.e., the GPIO mapping)

Property	Capsule	GPIO
Trig	DetectionSensor	GPIO 23
pinEcho	DetectionSensor	GPIO 24
rightMotorBackwards	EngineController	GPIO 5
leftMotorForward	EngineController	GPIO 22
rightMotorForward	EngineController	GPIO 6
leftMotorBackwards	EngineController	GPIO 27

Configuration page

Properties

Property	Value
Info	
derived	false
editable	true

Conclusion



Several customizations have been done for RTE system development within Papyrus-RT.

Some are general improvement :

- Shortening “Workflows” ;
- Simplifying the modeling of behaviors.

Others are specific to RTE systems :

- Animation view ;
- Monitoring view ;
- Configuration Page.

Work to do:

- Improve the action semantics ;
- Provide a generic implementation for loading platform libraries and configuration pages.

Tutorial @ MODELS 2016

MODELS 2016 Program - Chromium

MODELS 2016 Program x

program.models2016.irisa.fr

MODELS 2016 - Program

Sunday, Oct 2, 2016

Sessions	Rotonde Surcouf	Rotonde J. Cartier	Bouvet 1	Bouvet 2	Charcot	Vauban 1	Va
09:00			Doctoral Symposium ☆	Advanced Model Management with Epsilon ☆	ARCADIA in a Nutshell ☆	Hybrid Graphical/Textual Modelling and Code Generation with PapyrusRT ☆	ME ☆
10:30	Coffee Break						
11:00			Doctoral Symposium ☆	Advanced Model Management with Epsilon ☆	ARCADIA in a Nutshell ☆	Hybrid Graphical/Textual Modelling and Code Generation with PapyrusRT ☆	ME ☆
12:30		Lunch					

Thank You